الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE SYSTÈME D'EXPLOITATION

SÉRIE 06

OBJECTIFS PÉDAGOGIQUES :

À la fin de cette série, le stagiaire doit être capable de **c**onnaître :

- L'organisation en mémoire des fichiers
- La structure des fichiers
- Les différents SGF
- Les différents mécanismes d'allocation et de synchronisations des processus.

PLAN DE LA LEÇON :

INTRODUCTION

I- CATALOGUE EN MÉMOIRE

II- ARCHITECTURE DU SYSTÈME DE FICHIERS

1- Structure de fichiers
2- Méthodes d'accès
3- Contrôle des droits d'accès
4- Verrouillage des fichiers
5- Attribution des blocs

III- LES DIFFÉRENTS TYPES DE SGF

1- Le système de gestion de fichiers FAT
2- Le système de gestion de fichiers NTFS

IV- OUVERTURE ET FERMETURE DES FICHIERS

V- LES MÉCANISMES D'ALLOCATION

VI- LES MÉCANISMES DE SYNCHRONISATION

INTRODUCTION :

Un **système de gestion de fichiers** (SGF) est une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, disquette, CD-ROM, clé USB, disques SSD , etc.). :

Une telle gestion des fichiers permet de traiter et de conserver des quantités importantes de données ainsi que de les partager entre plusieurs programmes informatiques. Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.

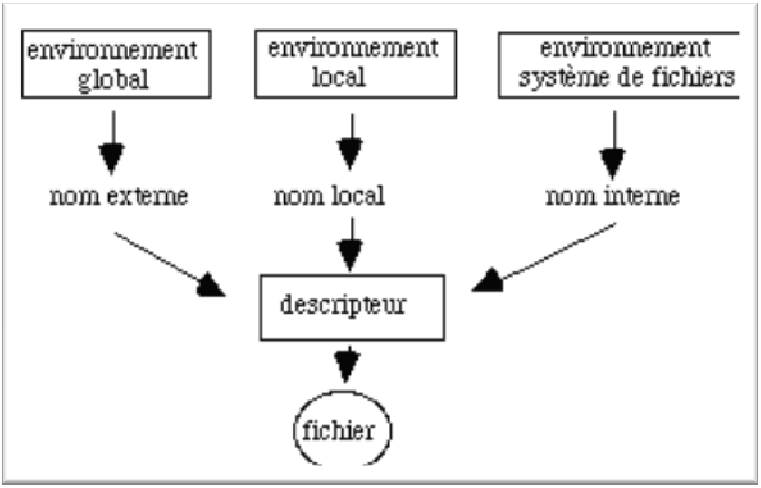
I- CATALOGUE EN MÉMOIRE :

Les fichiers sont regroupés dans des unités logiques appelées système de fichiers. Il correspond à un espace physique qui s'étend sur une partie d'un disque ou plusieurs disques. Pour l'utilisateur il apparaît comme un disque C:, D: ... sous Windows, avec un nom choisi à sa création pour Unix.

Le fichier est l'unité de conservation de l'information. C'est un objet complexe qui n'est pas constitué uniquement des informations visibles par l'utilisateur. Son descripteur contient les informations nécessaires à sa localisation physique sur le disque et à son usage. Le nom de ce descripteur est interne, c'est à dire que l'utilisateur ne le connaît pas. Il n'accède au fichier que par son nom externe. La liaison entre ces deux noms est établie au moyen du catalogue.

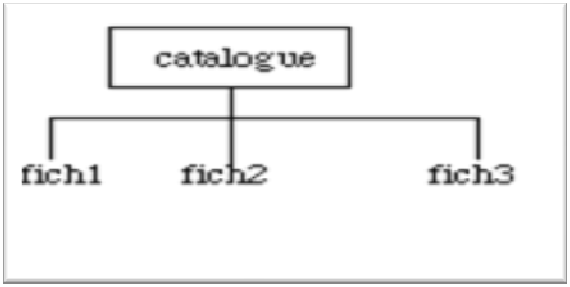
Un fichier peut éventuellement être défini par un nom local à la procédure en cours d'utilisation. C'est ce qui est fait lors de l'appel à la primitive d'ouverture dans un langage de programmation. Cette méthode est plus efficace que l'usage du nom externe car son interprétation est plus rapide et peut apporter une plus grande souplesse dans la gestion du fichier.

Ces différentes possibilités sont résumées dans la figure suivante **:**



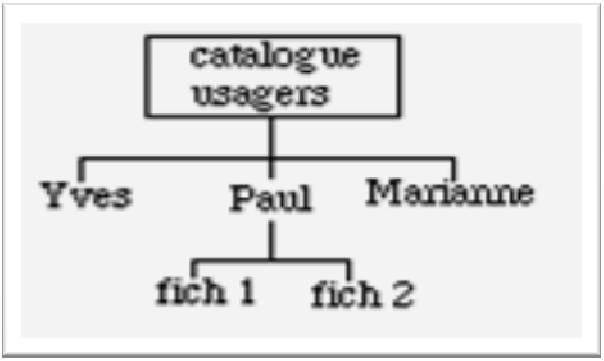
On rencontre le plus souvent trois modèles de catalogue:

- **Organisation en un seul niveau** : on associe directement le descripteur aux noms externes. Il n'y a aucun classement possible aussi ce système n'existe pratiquement plus.



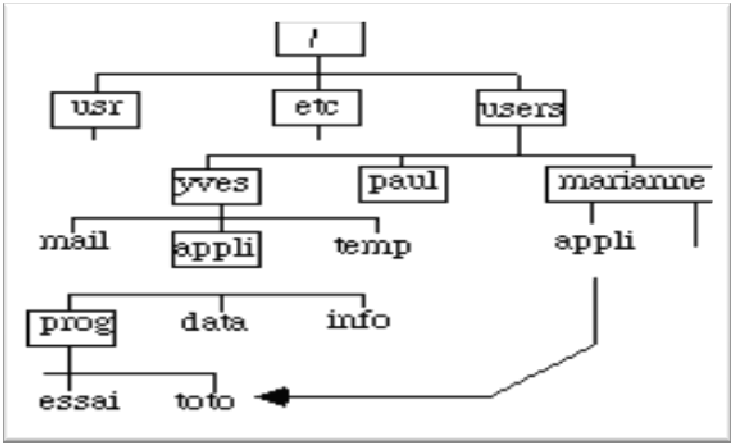
- **Organisation à deux niveaux**: il existe des catalogues secondaires par l'utilisateur. Certains systèmes comme ceux propres à IBM emploient des structures à un niveau qui ressemblent aux structures à deux niveaux. Les règles d'écriture dans le catalogue imposent un nom à plusieurs champs séparés par des points, le premier étant l'identificatif de l'utilisateur. Les

mécanismes de sécurité interdisent à une personne de retrouver les éléments du catalogue qui ne correspondent pas à ses propres fichiers, sauf autorisation explicite.



❖ **Organisation arborescente** :

C'est la plus connue de nos jours puisqu'elle est utilisée par Unix et les systèmes qui s'en sont inspirés (DOS puis Windows...).



II-ARCHITECTURE DU SYSTÈME DE FICHIERS :

1.1- Structure de fichiers :

Les fichiers peuvent être structurés de deux manières sous formes de suites d'octets non structurés ou d'une suite d'enregistrements. Un fichier est une séquence d'enregistrements de longueur fixe qui ont la même structure interne. Un fichier prend la forme d'un arbre d'enregistrements qui ne sont pas nécessairement de même longueur. Un enregistrement est une collection logique d'informations (par exemple, une ligne de texte, des informations relatives à une personne). Les opérations dE/S s'effectuent généralement en termes d'enregistrements. Le système d'exploitation peut gérer des structures d'enregistrements fixes et/ou variables. Les enregistrements peuvent à leur tour être divisés en champs, un champ représentant une donnée élémentaire (telle que le nom et l'âge). La figure suivante décrit la structure logique d'un fichier.

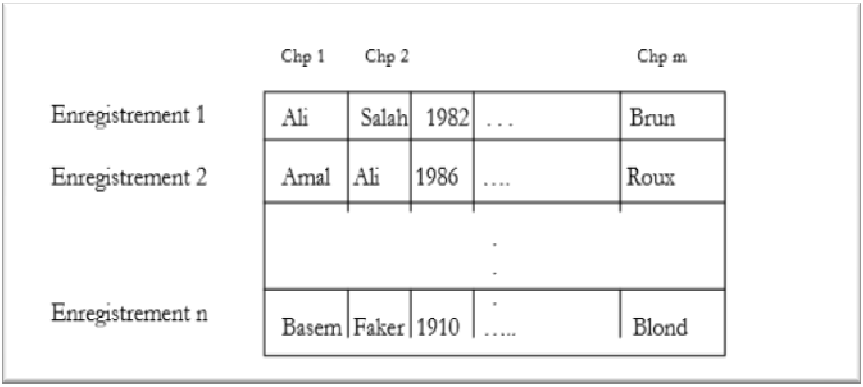


Figure 1: Structure logique d'un fichier

Transparent à l'utilisateur, le système d'exploitation considère un fichier comme une collection de blocs logique à taille fixe. Un bloc est l'unité de base d'une opération dE/S entre le disque et la mémoire tampon du système de fichiers. Le disque en tant que tel est un ensemble de blocs physiques. Chacun d'entre eux stocke un bloc logique et éventuellement d'autres données administratives. La taille du bloc est un multiple de l'unité d'E/S de base fournie par le pilote du disque.

1.2- Méthodes d'accès :

Il existe deux méthodes fondamentales pour accéder à des informations au sein d'un fichier **:**

Séquentielles et directes. Dans l'accès séquentiel, il faut accéder aux informations du fichier dans l'ordre dans lequel elles ont été stockées dans le fichier. L'accès se déroule de manière séquentielle depuis le début jusqu'à la fin. Les opérations de lecture ou d'écriture sur le fichier n'ont pas besoin de spécifier l'emplacement logique au sein du fichier, car le système d'exploitation maintient un pointeur de fichier qui détermine l'emplacement du prochain accès.

Avec l'accès direct, il est possible d'accéder à tout emplacement logique à l'intérieur du fichier. Généralement, l'accès direct peut être réalisé de 2 manières : Soit en spécifiant l'emplacement logique auquel accéder comme un paramètre à l'opération de lecture ou d'écriture, soit en spécifiant l'emplacement d'une opération de positionnement pour qu'il soit appelé avant la lecture ou l'écriture.

Les systèmes de base de données utilisent deux opérations d'accès au système d'exploitation élémentaire pour mettre en œuvre un large éventail de méthodes d'accès de haut niveau. Certains systèmes d'exploitation mettent en œuvre des méthodes d'accès de haut niveau par eux-mêmes. Parmi toutes ces méthodes, l'accès indexé est peut-être le plus significatif. Avec ce dernier, chaque enregistrement de fichiers dispose d'un ou plusieurs champs. Un champ sert de champ d'indexe. Les opérations de lecture et d'écriture comprennent un paramètre d'index. L'enregistrement avec la valeur d'index correspondante est l'enregistrement sur lequel est effectuée l'opération.

Pour toute méthode d'accès, les opérations de lecture et d'écriture peuvent être synchrones ou asynchrones. Les blocs d'E/S synchrones bloquent le processus jusqu'à la fin de l'opération d'E/S. Les E/S asynchrones renvoient immédiatement le contrôle au processus, laissant le processus libre de continuer à s'exécuter pendant l'E/S. Si les opérations d'entrées sont asynchrones, il faut faire appel à certains mécanismes pour avertir le processus que l'opération est

terminée. Pour cela, il est possible d'envoyer un signal au processus, d'attribuer une valeur particulière aux variables du processus ou de lancer un appel système spécial pour tester l'état de l'opération d'E/S. Les opérations de sortie asynchrones peuvent recourir aux mêmes techniques de notifications ou n'offrir aucune notification.

L'E/S synchrone représente la norme qui simplifie considérablement la programmation d'application.

La grande vitesse des opérations d'E/S de fichier n'incite en rien à l'utilisation des E/S asynchrones. Cependant ces dernières peuvent parfois être employées avec un périphérique d'E/S.

1.3- Contrôle des droits d'accès :

Le contrôle des droits d'accès établit une limite quant aux personnes pouvant accéder aux fichiers et à la manière dont elles peuvent y accéder. Le mécanisme de contrôle des droits d'accès le plus simple attribut un accès illimité à tous les utilisateurs. Il s'agit là du modèle de contrôle d'accès choisi par DOS. Sur un tel système, les utilisateurs qui souhaitent contrôler l'accès à leurs fichiers doivent mettre en place une limite d'accès physique (et sur le réseau) de leur machine.

Un aspect important du contrôle des droits d'accès est le type d'opérations à réaliser sur le fichier. Les opérations contrôlées comprennent entre autre **:**

- **La lecture**: lecture des informations contenues dans le fichier.
- **L'écriture :** écriture de nouvelles informations dans un fichier ou écrasement des informations d'un fichier.
- **L'adjonction :** écriture de nouvelles informations à la fin du fichier seulement.
- **La suppression :** suppression d'un fichier et libération de son espace de stockage en vue d'une utilisation dans d'autres fichiers.

- **La liste :** lecture des noms contenus dans un répertoire.
- **L'exécution :** chargement du contenu d'un fichier dans la mémoire principale et création d'un processus pour l'exécuter.
- **Le changement des droits d'accès :** modifications de certains droits d'accès d'utilisateur en vue d'une opération de contrôle.

L'autre caractéristique majeure du contrôle des droits d'accès est la manière dont il détermine ou non d'octroyer l'accès. Le mécanisme le plus commun consiste à baser la décision sur l'identité de l'utilisateur. Sur un système qui utilise une liste des droits d'accès, le système d'exploitation associe à chaque fichier le type d'opérations autorisé à chaque utilisateur. Dans un modèle de droits d'accès illimité, un ensemble indépendant de permissions est conservé pour chaque utilisateur, ce qui représente un volume important de données.

Un mécanisme de contrôle des droits d'accès limité réduit ce volume en regroupant les permissions d'accès pour un certain nombre d'utilisateurs ou de fichiers. Ainsi, de nombreux systèmes d'exploitation mettent en œuvre la notion de groupes d'utilisateurs. Chaque utilisateur et chaque fichier sont associés à un ou plusieurs groupes d'utilisateur. Au lieu d'avoir un ensemble de permissions d'accès pour chaque utilisateur, le fichier possède uniquement un ensemble de permissions d'accès pour son propriétaire et pour chaque groupe auquel il est associé. Tous les utilisateurs d'un groupe partagent les mêmes permissions d'accès. Un autre groupement fréquemment utilisé consiste à demander à ce que tous les fichiers d'un répertoire partagent les mêmes permissions.

L'autre base communément utilisée pour contrôler l'accès est le mot de passe. Pour réaliser une opération sur un fichier, un utilisateur doit spécifier le mot de passe de fichier associé à cette opération (un mot de passe de fichier est distinct de tout éventuel mot de passe d'ouverture de session). Comme avec les listes de droits d'accès, le groupement peut réduire le volume des données maintenues par le

système d'exploitation (et réduire le nombre de mots de passe que doit retenir un utilisateur). Ainsi, tous les fichiers d'un répertoire peuvent partager le même mot de passe.

Les capacités constituent une variante intéressante des listes de droits d'accès, mais cependant moins répandue. Sur les systèmes basés sur les capacités, les droits d'accès, au lieu d'être associés aux fichiers, sont associés aux processus. Lorsqu'un accès au fichier est tenté, le système d'exploitation vérifie le droit correspondant au fichier dans les droits d'accès associés au processus.

1.4-Verrouillage des fichiers :

Le verrouillage des fichiers offre aux processus la possibilité de mettre en œuvre un accès exclusif à un fichier. Trois grandes options existent dans la mise en œuvre du verrouillage :

- Le verrouillage peut se limiter à l'ensemble des fichiers ou la mise en œuvre peut autoriser de verrouiller certaines parties d'un fichier.
- Le verrouillage peut s'appliquer à tout accès ou il peut exister différents niveaux. Sur certains systèmes, il y à la fois des blocs de lecture et d'écriture. Un fichier verrouillé pour la lecture n'empêche pas un autre accès en lecture, mais l'accès en écriture par d'autre processus est refusé.
- Le verrouillage peut être soit obligatoire soit consultatif. Avec le verrouillage obligatoire, le système d'exploitation refuse l'accès pendant que le fichier est verrouillé. Avec le verrouillage consultatif, les primitives de verrouillage fournissent uniquement des informations sur l'état de verrouillage du fichier.
- L'accès n'est restreint que si un processus vérifie l'état de verrouillage du fichier et respecte le verrouillage qui est indiqué.

Dans certaines circonstances, un système d'exploitation peut mettre en œuvre un mécanisme de verrouillage implicite.

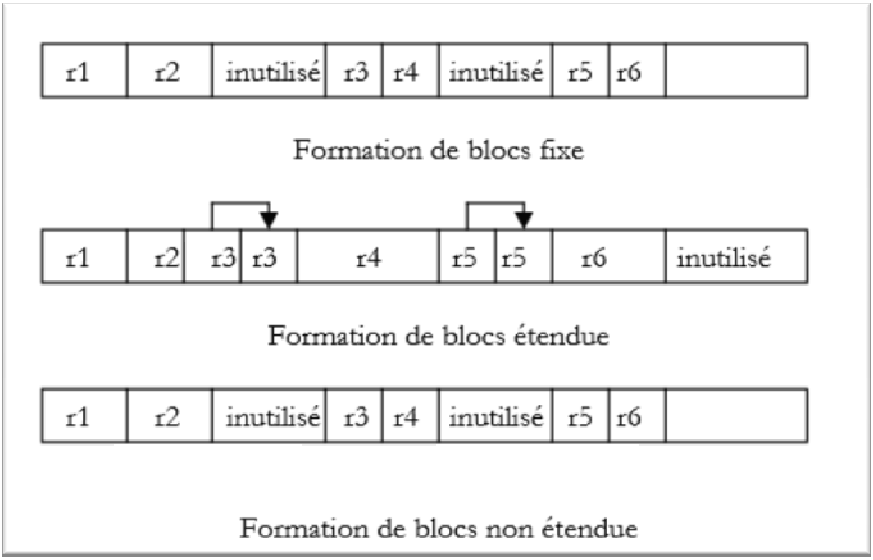
1.5- Attribution des blocs :

La méthode d'attribution de blocs détermine comment les enregistrements d'un fichier sont attribués dans des blocs.

Attribution fixe des blocs : pour les fichiers avec des enregistrements de taille fixe, un nombre intégral d'enregistrements est stocké dans chaque bloc. Aucun enregistrement ne peut être plus grand qu'un bloc. Si la taille du bloc n'est pas un multiple de la taille de l'enregistrement, il y aura de l'espace inoccupé à la fin du bloc. Le système d'exploitation peut calculer le bloc et l'offset à l'intérieur du bloc de tout enregistrement, en fonction de la taille de l'enregistrement et du bloc.

Attribution non étendue de blocs : pour les systèmes avec des enregistrements de taille variable, plusieurs enregistrements peuvent être stockés dans chaque bloc, mais aucun ne peut s'étendre sur plusieurs blocs. Les enregistrements ne peuvent pas non plus être plus grands que la taille du bloc. L'espace à la fin d'un bloc est gaspillé si le prochain enregistrement est plus important que cet espace. Le système d'exploitation ne peut calculer l'emplacement d'un enregistrement, à mains qu'il ne connaisse la taille de tous les enregistrements qui le précède.

Attribution étendue de blocs : les enregistrements peuvent être stockés dans plusieurs blocs. Il n'existe aucune limite quant à la taille d'un enregistrement et il n'y a pas d'espace inutilisé à l'intérieur d'un bloc. La seule manière de calculer l'emplacement d'un enregistrement d'un enregistrement consiste à additionner la taille de tous les enregistrements qui le précède



III- LES DIFFÉRENTS TYPES DE SGF :

Le SGF est dépendant du système d'exploitation. D'une manière générale, plus le système d'exploitation est récent plus le nombre de systèmes de fichiers supportés sera important.

Système d'exploitation	Type de système de fichiers supportés
DOS	FAT 16
Windows 95	FAT 16, FAT 32 (selon la version de Windows 95)
Windows 98	FAT 16, FAT 32 (selon la version de Windows 98)
Windows NT4	NTFS
Windows 2000 et XP	FAT 32, NTFS
MacOS	HFS
Linux / Unix	EXT2, ReiserFS

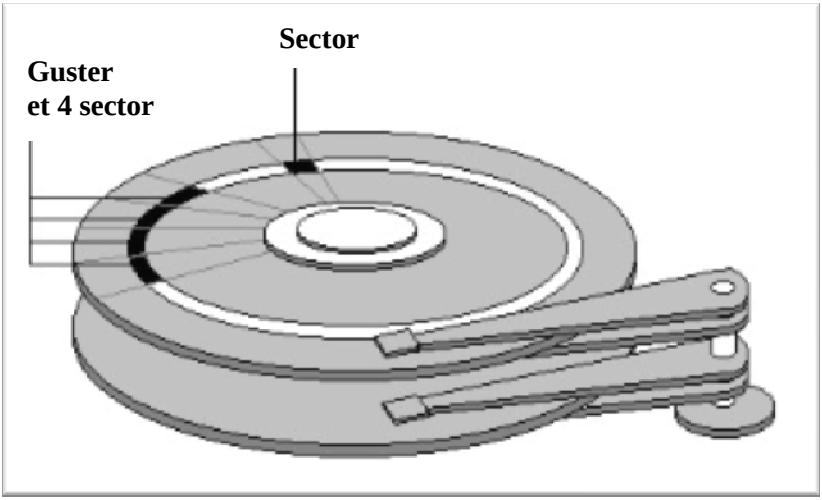
1- Le système de gestion de fichiers FATl :

Il comprend les éléments suivants **:**

- Un enregistrement d'amorçage principal avec la "Table des partitions" (MBR),
- Une zone réservée au secteur de chargement (Boot Loader),
- Un exemplaire de la FAT,
- Une copie optionnelle de la FAT,
- Le répertoire principal ou répertoire racine,
- La zone des données et sous-répertoires.

Le secteur de démarrage (MBR) (cylindre 0, tête 0 et secteur 1) contient la table de partitions et le code qui, une fois chargé en mémoire, va permettre d'amorcer le système (booter).

Ce programme, une fois en mémoire, va déterminer sur quelle partition le système va s'amorcer, et il va démarrer le programme (appelé boot strap) qui va amorcer le système d'exploitation présent sur cette partition.



D'autre part, c'est ce secteur du disque qui contient toutes les informations relatives au disque dur (fabricant, numéro de série, nombre d'octets par secteur, nombre de secteurs par cluster,…) Ce secteur est le secteur le plus important du disque dur, il sert au set up

du BIOS à reconnaître le disque dur. Ainsi, sans celui-ci le disque dur est inutilisable, c'est donc une des cibles préférées des virus.
Ce secteur de démarrage est appelé le boot manager sous Windows NT.

Les répertoires stockent toutes les informations sur chaque fichier qu'ils contiennent :

- Le nom de fichier ;
- La taille du fichier ;
- La date et l'heure de la dernière modification du fichier;
- Les attributs du fichier (lecture seule, archive, …);
- Le numéro du cluster auquel le fichier commence(les autres clusters constitutifs étant retrouvés par la FAT;
- Le répertoire parent (pour les répertoires autre que racine).

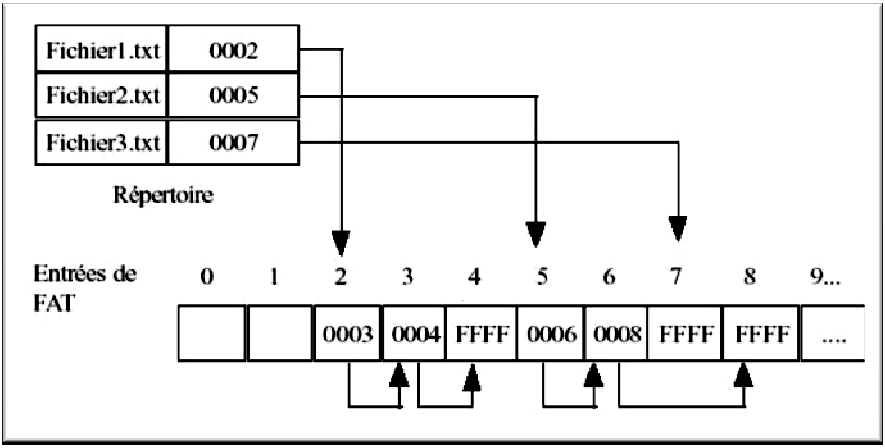
Le système de gestion de fichier FAT utilise un répertoire racine (représenté sur les systèmes d'exploitation qui utilisent ce type de systèmes de fichiers par le signe C:\), qui doit être situé à un endroit spécifique du disque dur.

Ce répertoire racine stocke les informations sur les sous-répertoires et fichiers qu'il contient.

Le système de gestion de fichier FAT est aussi caractérisé par l'utilisation d'une table d'allocation de fichiers et de clusters (ou blocs).

La FAT (File Allocation Table: table d'allocation des fichiers) est le cœur du système de gestion de fichiers. Elle est localisée dans le secteur 1 du cylindre 0 à la tête 1

La Table d'Allocation de Fichiers est une liste de valeurs numériques permettant de décrire l'allocation des clusters d'une partition, c'est-à-dire l'état de chaque cluster de la partition dont elle fait partie.



La table d'allocation est en fait un tableau dont chaque case (ou entrée) correspond à un cluster. Chaque case contient un chiffre qui permet de savoir si le cluster qu'elle représente est utilisé par un fichier, et, le cas échéant, indique l'emplacement du prochain cluster que le fichier occupe.

On obtient donc une chaîne FAT, c'est-à-dire une liste chaînée de références pointant vers les différents clusters successifs, jusqu'au cluster de fin de fichier. Chaque entrée de la FAT a une longueur de 16 ou 32 bits (selon qu'il s'agit d'une FAT16 ou d'une FAT32).

Les deux premières entrées permettent de stocker des informations sur la table elle-même, tandis que les entrées suivantes permettent de référencer les clusters.

Certaines entrées peuvent contenir des valeurs indiquant un état du cluster spécifique. Ainsi la valeur 0000 indique que le cluster n'est pas utilisé, FFF7 permet de marquer le cluster comme défectueux pour éviter de l'utiliser, et les valeurs comprises entre FFF8 et FFFF spécifient que le cluster contient la fin d'un fichier. Chaque partition contient en réalité deux copies de la table, stockées de manière contiguë sur le disque, afin de pouvoir la récupérer si jamais la première copie est corrompue. Derrière la copie de la fat commence le répertoire principal.

2- Le système de gestion de fichiers NTFS :

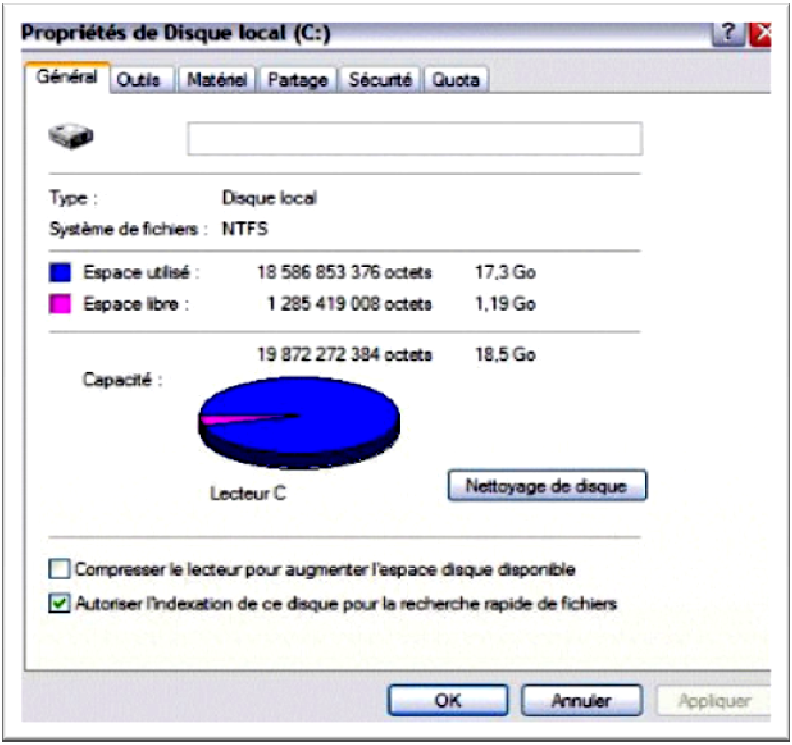
NTFS (New Technology File System) utilise un système basé sur une structure appelée « table de fichiers maître », ou MFT (Master File Table), permettant de contenir des informations détaillées sur les fichiers. Ce système permet ainsi l'utilisation de noms longs, mais, contrairement au système FAT32, il est sensible à la casse, c'est-à-dire qu'il est capable de différencier des noms en majuscules de noms en minuscules.

Pour ce qui est des performances, l'accès aux fichiers sur une partition NTFS est plus rapide que sur une partition de type FAT.

La limite physique d'un disque est de 2To. Les noms des fichiers peuvent comporter jusqu'à 255 caractères.

C'est au niveau de la sécurité que NTFS prend toute son importance, car il permet de définir des attributs de sécurité pour chaque fichier (droits associés aux utilisateurs …).

La version 5 de ce système de fichiers (en standard sous Windows 2000) amène encore de nouvelles fonctionnalités parmi lesquelles des performances accrues, des quotas de disque par volume définis pour chaque utilisateur.



❖ La table des fichiers maîtres : MFT

La FAT est un tableau de valeurs numériques dont chaque case permet de décrire l'allocation des clusters d'une partition, c'est-à-dire l'état (l'occupation ou non par un fichier) de chaque cluster de la partition dont elle fait partie.

Le système de fichiers NTFS est basé sur une structure différente, appelée table de fichiers maître, contenant des enregistrements sur les fichiers et les répertoires de la partition.

Le premier enregistrement, appelé descripteur, contient des informations sur la MFT (une copie de celui-ci est stockée dans le second enregistrement).

Le troisième enregistrement contient le fichier journal, un fichier qui contient es actions effectuées sur la partition.

Les enregistrements suivants, constituant ce que l'on nomme le noyau, référencent chaque fichier et répertoire de la partition sous forme d'objets affectés d'attributs.

Cela signifie que les informations relatives à chaque fichier sont stockées dans le fichier, qui est lui même enregistré au sein de la MFT. La MFT représente donc une structure de stockage des données de la partition, et non une liste de clusters.

IV- OUVERTURE ET FERMETURE DES FICHIERS :

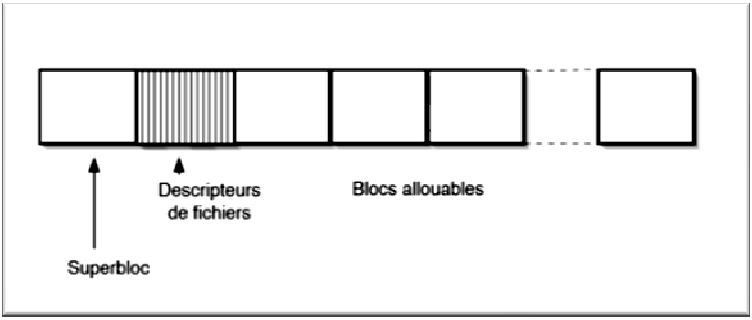
À l'ouverture d'un fichier par un processus, l'appelant fournit un nom symbolique absolu ou relatif et l'appel système lui retourne un numéro de descripteur de fichier ouvert ou un code d'erreur. Le numéro de descripteur sera utilisé comme désignation locale au processus pour toute la phase pendant laquelle le fichier reste ouvert.

C'est à l'ouverture que sont effectuées toutes les vérifications nécessaires (nom correspondant à un fichier inexistant par exemple) et que sont construites les différentes tables permettant de réaliser les accès. L'appel système correspondant est

ouvrir (nom_fich,mode) ->i_dfo

Lorsque l'ouverture se passe bien, ouvrir retourne un indice de descripteur de fichier ouvert.

En cas d'erreur, ouvrir retourne un code d'erreur négatif avec le sens suivant **:**



1- On devrait avoir un catalogue et on traverse un fichier,
2- Nom inexistant dans un catalogue,
3- Trop de fichiers ouverts dans le système,
4- Trop de fichiers ouverts par le processus.

L'appel système de fermeture reçoit en paramètre un numéro de descripteur de fichier ouvert.

Les tables construites à l'ouverture sont détruites. L'appel système est **:**

Fermer(i_dfo) -> code de retour

La procédure retourne **:**

- 0 si l'opération s'est bien exécuté,
- 1 si le descripteur est invalide,
- 2 s'il ne correspond pas à un fichier ouvert.

V- LES MÉCANISMES D'ALLOCATION :

Il existe 3 modèles de base pour allouer l'espace de l mémoire auxiliaire à des fichiers. Le schéma d'allocation est chargé d'associer des blocs logiques d'un fichier à des blocs physiques de la mémoire auxiliaire.

Dans la plupart des systèmes d'exploitation, la taille d'un bloc physique est une puissance de 2 comprise entre 512 et 4096.

1- Allocation contiguë :

Le modèle le plus simple est l'allocation contiguë. Les blocs logiques d'un fichier sont stockés dans une partition de blocs physiques contigus. L'entrée du répertoire a uniquement besoin de stocker l'adresse de la mémoire auxiliaire de départ du fichier ainsi que la taille de ce dernier. L'emplacement physique de tout octet du fichier peut être calculé en ajoutant l'offset approprié à l'adresse de la mémoire auxiliaire de départ de fichier.

Lorsqu'un fichier est créé, l'allocation contiguë requiert une pré-allocation d'espace pour le fichier. Le système d'exploitation peut être conçu pour étendre l'allocation du fichier, si nécessaire. Si l'expansion est autorisée et que l'espace de stockage au-dehors de la fin du fichier est utilisé, un ou plusieurs fichiers doivent être déplacés pour loger le fichier le plus important. Les fichiers à déplacer doivent recevoir de nouvelles partitions sur le disque, puis être copiés sur ces nouvelles partitions.

2- Allocation chainée :

Dans l'allocation chainée, les blocs physiques dans lesquels est stocké un fichier peuvent être dispersés dans l'ensemble de la mémoire auxiliaire. Les blocs physiques sont plus importants que les blocs logiques et stockent à la fois le bloc logique et un pointeur vers le bloc physique dans lequel est stocké le prochain bloc logique du fichier. L'entrée du répertoire stocke l'emplacement du premier bloc physique. Le bloc physique associé au Nème bloc logique peut être

déterminé uniquement en lisant les précédents blocs N-1 et en suivant les liens qu'ils contiennent. Les performances des opérations d'adjonctions (écriture à la fin d'un fichier) peuvent être améliorées de façon significative en incluant également dans l'entrée du répertoire un pointeur vers le dernier bloc de la file.

3- Allocation indexée :

L'allocation indexée est une variante de l'allocation chainée. Le bloc physique stocke seulement le bloc logique, qui est par conséquent de la même taille qu'un bloc logique. Les liens vers les blocs physiques d'un fichier sont stockés de manière contiguë dans une table d'index. L'entrée du répertoire contient soit la table d'index soit un pointeur vers celle-ci.

Avec les allocations contiguës et chainées, il suffit au système de fichiers de stoker l'emplacement physique de départ du fichier. Toutes les autres adresses peuvent être déterminées à partir de l'emplacement de départ du fichier. Avec l'allocation indexée, le système de fichiers doit avoir une entrée d'index pour chaque bloc du fichier. Pour minimiser le volume d'espace requis dans les structures de répertoire, l'indexation à plusieurs niveaux peut être utilisée.

VI- LES MÉCANISMES DE SYNCHRONISATION :

On peut les classer en deux catégories :

1- Les mécanismes simples qui se limitent à des échanges de signaux de marche et d'arrêt, analogues à des feux rouges. Ces signaux sont divers : sémaphores, signaux Unix...

2- Des mécanismes plus complexes qui permettent également d'échanger des informations ou messages.

1- Synchronisation par moniteur et sémaphores :

a- Synchronisation par moniteur :

Le moniteur est le cœur du système d'exploitation. Il est constitué d'un ensemble de procédures et de variables d'état utilisées par ces procédures. Le moniteur représente le cœur du système d'exploitation. Il échappe aux règles habituelles des processus car il est toujours présent et son rôle est d'ordonnancer leur fonctionnement.

Certaines de ses variables sont accessibles à l'utilisateur grâce à des bibliothèques de fonctions spéciales au travers de points d'entrée du moniteur. Les processus externes au moniteur peuvent interroger et utiliser ces variables mais elles ne peuvent pas les modifier car il est fort probable que cela perturberait le fonctionnement du moniteur donc de l'ordinateur.

Parmi ces fonctions il en existe qui permettent de bloquer ou de réveiller les processus écrits par les utilisateurs. Le blocage et le réveil s'effectuent au moyen d'une condition **c** utilisable dans les trois opérations suivantes:

- **attendre(c):** Bloque le processus p qui l'utilise et le place en attente de l'événement c.
- **vide(c) :** Retourne vrai s'il n'existe pas de processus en attente de c, faux sinon.
- **Activer(c) :** Réveille le premier processus en attente de l'événement c.

On notera que la fonction attendre(c) suppose l'existence d'une file d'attente associée. Le processus réveillé reprend son activité à l'instruction qui suit le point d'arrêt.

Une caractéristique essentielle de la synchronisation par moniteur: les processus consultent l'état de variables qu'ils ne peuvent modifier en aucun cas. C'est donc un moyen assez frustre car les communications sont réduites au minimum mais simple à réaliser. Ces mécanismes existent à des degrés divers dans tous les systèmes d'exploitation ; il est possible de bloquer le père au moyen d'un appel à une fonction wait().

b- Synchronisation par sémaphore :

La synchronisation par sémaphore ou flag est un moyen simple et ancien de synchroniser des processus parallèles. A la différence de la synchronisation par le moniteur le programmeur a accès aux états de ce drapeau et peut le manipuler. Le principe est directement hérité des chemins de fer d'où son nom : lorsque le sémaphore est levé, le processus P peut continuer son chemin; lorsqu'il est baissé il doit attendre jusqu'à ce qu'un autre processus Q le lève. P et Q sont l'équivalent de deux trains roulant sur deux voies distinctes qui doivent se synchroniser pour pouvoir franchir un croisement sans accident.

Dans le modèle le plus simple il existe trois primitives:

- **Lever (c) :** fait passer le sémaphore c de la valeur "baissé" à "levé".
- **Baisser(c) :** fait passer le sémaphore c de "levé" à "baissé".
- **Flag(c) :** retourne vrai si le sémaphore c est baissé.

Les sémaphores sont des variables communes mises à la disposition des différents processus par le système d'exploitation qui peuvent les consulter, les modifier et sur lesquelles ils peuvent se mettre en attente.

Pour l'illustrer voici un exemple du transfert de résultats au moyen d'un tampon:

Processus P	Processus Q

baisser(e);
while (calculs à faire){
calculs des éléments du tampon (a);
baisser(e);
if (flag(f)) {
attendre(f);
}
écrire tampon (a);
lever(e);
	lever(f);
	while (éléments à transférer) {
	if (flag(e)) {
	attendre(e)
	}
	baisser(f);
	lire tampon(a);
	lever(f);
	}

e et f sont des variables qui pouvaient être communes à deux processus distincts.

À la différence des threads les segments de données sont disjoints donc il faut imaginer un moyen de partager ces informations. Les sémaphores sont une solution pour résoudre ce problème. Ils font partie du contexte commun à l'ensemble des processus. Ces variables ne peuvent prendre que deux valeurs. Elles doivent être déclarées au moyen d'instructions spéciales qui précisent les noms des processus qui peuvent les partager. Les sémaphores sont un moyen simple de synchronisation qui, cependant, est en désuétude car il ne permet qu'un partage pauvre (binaire) d'informations.
On préfère aujourd'hui une synchronisation par messages qui, comme son nom l'indique, permet d'échanger des informations.

2- Synchronisation par messages :

❖ Principes généraux :

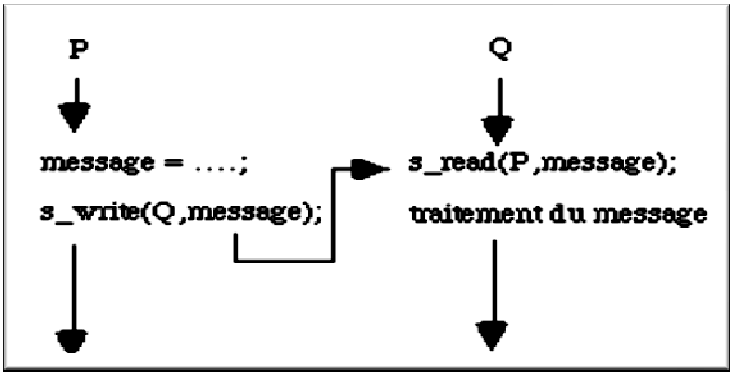
Le sémaphore est un moyen de communication frustre entre processus car limité à un message binaire : on passe ou on ne passe pas. On peut imaginer des primitives plus riches qui permettent la synchronisation en échangeant simultanément des informations appelées messages. Les primitives s'apparentent à des ordres de lecture et d'écriture :

- s_read (id,message) : met le processus en attente d'une lecture jusqu'à ce qu'un processus de nom id envoie un message. On l'appelle une lecture bloquante.

- s_write (id,message) : envoie au processus de nom id un message. Cette fonction n'est pas bloquante. Le processus qui l'utilise continue son exécution même si le processus receveur n'a pas lu le message. Celui-ci est stocké dans un tampon. Si plusieurs messages sont envoyés successivement à ce processus, leur ordre de lecture sera l'ordre d'émission des messages.

La synchronisation par messages est très utilisée dans les systèmes d'exploitation modernes car elle permet, grâce à l'information échangée, d'envisager les traitements les plus variés. Dans la réalité il existe de nombreuses autres primitives qui permettent de réaliser des fonctions complexes.

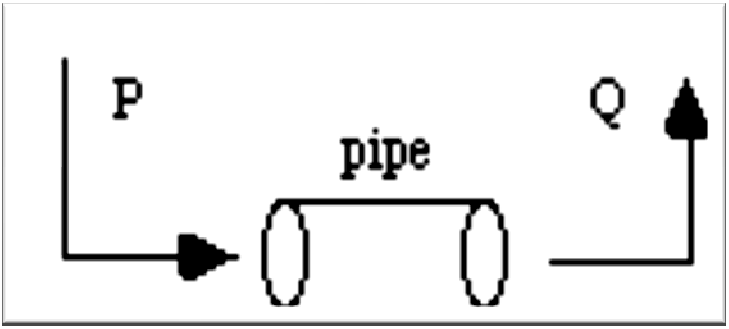
La synchronisation par message trouve aujourd'hui un usage encore plus large. Ainsi PVM et MPI sont des bibliothèques qui permettent de réaliser les fonctions d'échange, de synchronisation et de communication entre processus qui sont exécutés sur des machines en réseau. PVM et MPI sont employés sur les machines massivement parallèles.



Unix connaît un mécanisme de communication entre processus, appelé pipe, qui est un exemple de synchronisation possible par messages. Un premier processus P écrit dans le pipe, un deuxième, Q, le lit. La file d'attente correspond au modèle FIFO

Un fichier partagé est une autre méthode, très simple et efficace, pour faire communiquer et synchroniser deux processus: le premier crée et écrit dans un fichier, le deuxième en lit le contenu. On peut également utiliser l'existence ou la non-existence d'un fichier en guise de sémaphore. Ceci est souvent utilisé dans les systèmes Unix où l'on voit souvent se créer de nombreux fichiers dont l'existence est très temporaire, simplement pour permettre à un ensemble de processus de synchroniser leur activité et de communiquer l'information indispensable au bon déroulement de la fonction pour laquelle ils ont été créés.

Il se peut parfois, parce qu'un processus ne fonctionne pas correctement, que ce fichier ne soit pas détruit après usage. Il suffit alors de le détruire manuellement pour retrouver un fonctionnement correct. La synchronisation par fichiers est peu performante car les opérations de lecture et d'écriture sont lentes. Elle ne peut donc pas être employée lorsqu'on a besoin de temps de réaction courts.



EXERCICES D'APPLICATION :

1- Quelles sont les commandes qui permettent d'ouvrir et de fermer des fichiers?

2- Citez les différentes méthodes de synchronisation des processus ?

3- Qu'est ce qu'un cluster?

4- Comment se nomme l'unité minimale allouée par un disque dur lors d'une opération d'écriture ?

- **b-** Le Secteur.
- **c-** Le Cluster.
- **d-** La FAT.
- **e-** Le Block.

5- À quoi sert un système de fichier ?

- **a-** Il permet de stocker les informations et de les organiser sur la mémoire cache.
- **b-** Il permet de stocker les informations et de les organiser sur la mémoire vive.
- **c-** Il permet de stocker les informations et de les organiser sur les mémoires secondaires.

CORRECTIONS DES EXERCICES :

1 et 2 revoir le cours

- 3.

- Un cluster (ou unité d'allocation » ou bloc) est la plus petite unité de disque que le système d'exploitation est capable de gérer.

- Un fichier occupe un nombre entier de cluster.
- Un petit fichier, aussi petit soit il, occupe tout un cluster. Un fichier peut occuper plusieurs clusters, mais le dernier cluster ne sera pas rempli:

- Il y a un gaspillage de l'espace disque, d'autant plus grand que la taille des clusters est grande.

- Plus la taille d'un cluster est importante, moins le système d'exploitation aura d'entités à gérer et plus il sera rapide...
- En contrepartie, étant donné qu'un système d'exploitation ne sait gérer que des unités d'allocation entière, c'est-a dire qu'un fichier occupe un nombre entier de cluster, le gaspillage d'espace disque est d'autant plus grand que le secteur est grand.
- Il faut choisir une taille de cluster optimale pour ne pas trop ralentir l'OS et ne pas trop gaspiller d'espace disque.

4- b

5- c